Remarks:

Claims 1 and 3-4 were previously pending with claim 1 being independent. Claim 1 and 3 has been amended and claims 34-38 have been added, and therefore claims 1, 3-4, and 34-38 are currently pending with claims 1, 34 and 37 being independent.

Claim 1 has been amended to include the limitation "on a pervasive computing device" on several of the elements of the claim. This amendment is supported by ¶¶ 0045, 0047, 0051, 0128 of the specification.

Claim 3 has been amended to replace the term "pointer" with the term "reference". This change is supported because the terms refer to fundamentally the same concept. The primary difference is that the term "pointer" is largely specific to C or C++, whereas the term "reference" is used generally.

Claims 34-38 find support from original claims 14, 16-17, 24, 26, and 28, respectively.

The Rejection of Claims under 35 USC § 102

Claims 1 and 3 were rejected under 35 U.S.C. 102(e) as being anticipated by "Using JDeveloper to Build Oracle XML Applications" ("Oracle"). Oracle teaches methods for using XSQL and XML in Java applications using JDeveloper. Applicant respectfully disagrees that Oracle anticipates the current invention.

Oracle does not anticipate the current invention because Oracle does not disclose all the elements present in claim 1. Specifically, Oracle does not teach "on a pervasive computing device", "interpreting the queries by associating at least one declarative language function with the query

terms by converting the SQL to an intermediate tree representation corresponding to the declarative language function" and "converting the queries represented by the at least one declarative language function to a plurality of JAVA statements" as recited in claim 1.

Oracle Does Not Teach "On a Pervasive Computing Device . . ."

Oracle does not teach "managing a relational database on a pervasive computing device." In fact, Oracle teaches a system where no database management, other than submitting XSQL queries, is performed on the pervasive device. At page 1 of Oracle, the Business Components for Java (BC4J) component is described as "a standards-based, *server-side* framework" for building applications. This configuration is mirrored in Figure 11-2 of Oracle and on page 3 of Oracle. Oracle also uses a servlet to interpret XSQL to "input an XML file containing embedded SQL queries." Oracle p. 11. Servlets necessarily run within a Java Web/Servlet container ("Servlet Container"). Oracle does not specify where the XSQL Servlet is deployed or, similarly, where the Servlet Container resides. However, because the BC4J components, which can also be servlets, are deployed to a server, it is likely that XSQL Servlet would also be deployed to a server. In fact, Oracle does not imply that the pervasive computing device is used for anything other than submitting an XSQL query and receiving results as XML or HTML. Oracle Figure 11-8, pp. 18, 20 ("Shows an example code snippet. . . which transforms the XML data to HTML for displaying on a browser of the Palm Pilot emulator."). At best, Oracle teaches querying a database and displaying the results of the query on a pervasive device. Therefore, Oracle does not teach "managing a relational database on a pervasive computing device."

In contrast, the current application ("Grust") envisions "[a] method system, and program for *managing a relational database in a pervasive computing environment*." Grust at (57). "The runtime environment for [this invention] is characterized by extremely tight random access memory availability and relatively weak processing power (e.g., palm-sized PDAs, embedded devices)." Grust ¶ 0045. However, despite "this restricted computing environment, the database management system described herein is able to implement a relational database management system (RDMBS) service on these devices." Grust ¶ 0047. Additionally, "[t]he method, system, and program product of the invention enables a small footprint DBMS to offer a flexible SQL query interface for restricted runtime environments." Grust ¶ 0051. To that end, 'this invention enable[s] these runtime environments to efficiently interpret and evaluate programs in any language." Grust ¶¶ 0051, 0128.

While FIG. 1 and FIG.2 envision databases residing on servers, these embodiments do not exclude databases simultaneously residing on the pervasive device. For example in FIG. 2, the Handheld Device 18 communicates via Connect Services 13 to a Synchronization Server 11a that communicates to Databases 12a, 12b through several layers of servers 11b, 11c. The purpose of a *Synchronization Server* is to synchronize data that resides both on the Databases and the Handheld device. If no data resided on the Handheld Device, synchronization would not be necessary. Therefore, the pervasive device can "receiv[e] queries", "interpret[] the queries", "convert[] the queries", and "execut[e] the [converted queries]" without accessing the data on the server. Thus, the current invention claims "[a] method of managing a relational database *on a pervasive computing device*" which is not taught by Oracle.

Oracle Does Not Teach "Interpreting the Queries . . ."

Oracle does not teach "interpreting the queries by associating at least one declarative language function with the query terms by converting the SQL to an intermediate tree representation corresponding to the declarative language function" because the query terms are not associated with at least one declarative language function, the SQL is not converted to an intermediate tree representation corresponding to the declarative language function, and because XSQL is not a declarative language. Taking the broadest interpretation, a query term is some subset of an entire query. In light of the detailed description of the original Application, a query term is roughly analogous to a token and, at least, tokens representing queries, sub-queries, functions, and filter conditions are associated with a declarative language function. Thus the query from FIG. 6:

```
SELECT x,b FROM S,T WHERE y=a
```

Would likely be represented in a declarative language, such as LISP, as:

```
(select    (x, b)
           (from (table s) (table t))
           (where (equal y a)))
```

In this form, query terms, such as select, from, where, and "=", are associated with the declarative language functions select, from, where and equal, respectively. Furthermore, the structure of the function creates an intermediate tree representation that corresponds to the declarative language function, as required by claim 1. The tree representation is shown in FIG. 6 of the specification.

Oracle, by contrast, teaches embedding a whole SQL statement within an XML wrapper. There is no association of query *terms* with at least one declarative language function. Assuming for the moment that XSQL qualifies as a declarative language, there is only an association of the entire query with a declarative language function. In Example 1 on page 11 of Oracle, the *entire query* would be associated to the xsql:query function. In this form, the *SQL* is not converted into an intermediate tree structure corresponding to a declarative language function. Rather, the only tree structure corresponding to a declarative language function is in the *XML wrapper* around the SQL statement. In other words, the query of Example 1 can be represented as a tree with the FAQ node as the parent of the xsql:query node. However, even in this representation, the *SQL* is not converted into an intermediate tree structure corresponding to a declarative language function, but rather the XML wrapper is in a tree structure while the SQL is unchanged.

Additionally, in the section entitled "XML in Business Components for JAVA", also cited by the Examiner, XML is used to "define the metadata that represents the declarative language settings and features of the objects." Oracle does not describe the functionality imparted by BC4J. However, figure 11-2 of Oracle clearly shows that the BC4J component does not interact with the XSQL portion of the JDeveloper framework and, based on the arrow leading outward from the Oracle9i database to the BC4J object, the BC4J does not interact with the database queries at all, but only a set of rows that are the result of a query. If BC4J does not interact with queries, it cannot teach "interpreting the *queries* by associating at least one declarative language function with the query terms by converting the SQL to an intermediate tree representation corresponding to the declarative language function" (emphasis added) or "converting the *queries* represented by the at

least one declarative language function to a plurality of JAVA statements" (emphasis added).

XSQL is also not a declarative language as defined by the specification because SQL is not a declarative language and the addition of an XML wrapper does not convert a non-declarative language into a declarative language. In ¶ 0042, a declarative language is defined as one "written in the form of function calls, where the 'program' is a series of function definitions and function calls." SQL is not written in the form of function calls and does not declare function definitions. Therefore SQL does not conform to this definition and cannot be defined as a declarative language.

The XML wrapper shown in Example 1 on page 11 of Oracle does not convert a non-declarative language into a declarative language. In the Example 1, there is no explanation of XSQL other than a single query. With only one query and no additional explanation, there is insufficient information to demonstrate that XSQL is anything more than an XML wrapper around a standard SQL statement. In that light, XSQL cannot be a declarative language if SQL is not declarative. Because SQL is not a declarative language, XSQL is not.

Therefore, Oracle does not teach "interpreting the queries by associating at least one declarative language function with the query terms by converting the SQL to an intermediate tree representation corresponding to the declarative language function."

Oracle Does Not Teach "Converting the Queries. . ."

Additionally, Oracle does not teach "converting the queries represented by the at least one declarative language function to a plurality of JAVA statements" because the Oracle does not represent queries as declarative language functions and Oracle does not convert queries to a plurality

of Java statements.

Oracle does not teach "converting the queries represented by the at least one declarative language function to a plurality of JAVA statements" because the Oracle does not represent queries as declarative language functions, as discussed above. Oracle may instead define a wrapper for a query in a declarative language format. Because the "queries are not represented by at least one declarative language function", "queries represented by the at least one declarative language function" cannot be converted to "a plurality of JAVA statements."

Oracle also does not teach "converting the queries represented by the at least one declarative language function to a plurality of JAVA statements" because Oracle does not convert queries to a plurality of Java statements. To convert queries to a plurality of Java statements implies that, for every query, new Java code is created, the new Java code is compiled and the, new Java code is executed. Here, Oracle does not teach any of these. The XML Web Bean teaches "generat[ing an] XML containing the data from a View Object and render[ing] it to the output stream of a JSP response." There is nothing provided in Oracle which could lead to the assumption that the XML Web Bean creates new Java code, causes the new Java code to be compiled, and then causes the new Java code to be executed. In fact, the XML Web Bean is likely a precompiled java object which receives a request, possibly containing a XSQL query, and then either removes the XML wrapper and passes the SQL query to JDBC for execution, or simply provides the full XSQL query to another object for execution. In either case, the XML Web Bean does not create, compile or execute new Java code, rather it interprets requests as data and executes its own precompiled code to achieve the desired result. Therefore, Oracle also does not teach "converting the queries represented by the at

least one declarative language function to a plurality of JAVA statements."

Therefore, Applicant submits that the § 102 rejections are improper because Oracle does not disclose all the elements present in the current invention.

The § 103 rejection of claim 4 is dependent on Oracle disclosing all the elements of parent claim 1 which the Applicant respectfully contends that Oracle does not, for the reasons stated above.

For at least the reasons set forth above, applicant respectfully submits that claims 1, 3-4, and 34-38 are now in allowable condition and requests a Notice of Allowance. In the event of further questions, the Examiner is urged to call the undersigned. Any additional fee which is due in connection with this amendment should be applied against our Deposit Account No. 19-0522.

Respectfully submitted,

By: /Scott R. Brown/
Scott R. Brown, Reg. No. 40,535
HOVEY WILLIAMS LLP
10801 Mastin Blvd., Suite 1000
84 Corporate Woods
Overland Park, KS66210
(913) 647-9050

ATTORNEYS FOR APPLICANT(S)